Проект

„Задача за търговският пътник“

по

Структури от данни

Изготвил:Йордан Павлов FN№71978

Съдържание:

Глава 1.Увод

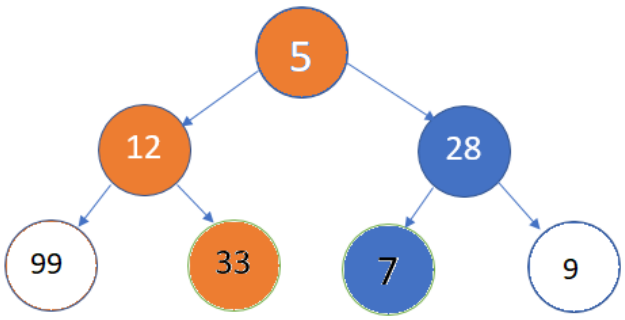
1.1 Търговският пътник има задачата да тръгне от определен град и да обиколи всички градове, който е предвидил за доставка на продукти и да се върне в града от където е тръгнал, като трябва да е мине по възможно най-краткият път. Проблемът е с NP сложност изисква проверка на резултатът. Евристиката представлява правене на извод, но няма гаранция ,че резултатът е оптимален.

1.2 Структурата на докоментацията се извършва по следните стъпки. Първата е да се представяне на належащите проблеми. Втората е търсенето на достатъчно ефективно решение. Документацията завършва с представянето на различни примери и сценарии с цел да се онагледи поведението на програмата в различните сценарии.

Глава 2.Алгоритми

1. Най-близък съсед:

Този метод е един от първите алгоритми за решаване на проблема за търговският пътник. Търговският пътник тръгва от произволен град и многократно посещава най-близкият град, докато не обиколи всички градове. Алгоритъмът е сравнително бърз , обикновено не е оптимален. Сравнително дава средно път с 25% по-дълъг път от възможно най-краткия път, тоест това не е най-доброто решение. Причината за този факт е алчността ,която се стреми да осигури бързи резултати ,а не преглежда по-големият набор от възможности. Един път взето решение, то никога не се преразглежда. Спестява стъпки пести и време на работа.



Начин на работа:

1. избиране на връх (зададен от потребителя).

2. винаги, когато достигне някакъв връх, поглежда тежестите на всички ръбовете, които водят до върхове, които все още не сте посетили и съответно избира този с най-малко тегло.

3. след като достигне последният връх се връща към началната си точка.

Скоростта на този алгоритъм е n^2.

2. 2-opt:

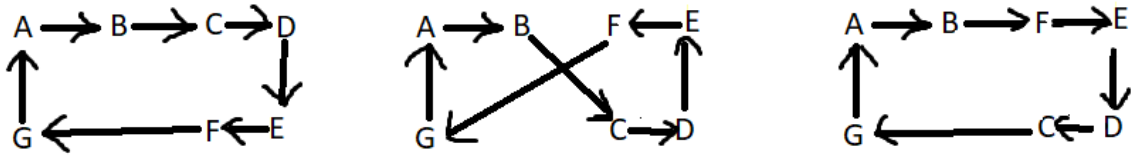
Този метод търсене на път със специално разменяне, което работи като негова евристика. Разменянето представлява избиране на 2 града примерно i и j премахваме двата пътя между i и предишният на i и j и следващият на j, като ги заменя с път между i и j, както между предишният на i и следващият на j , като проверява дали новополученият път е по-кратък. Ако пътят е по-дълъг продължаваме съответно без да премахваме пътища. Скоростта е (n-2)^2(log n). На графиката показвам ,че времето за работа на този метод е експотенциален, по-бавен е от най-близкият съсед, но този алгоритъм показва максимално най-краткият път с малко отклонение.

Пример за път: A => B => C => D => E => F => G => A

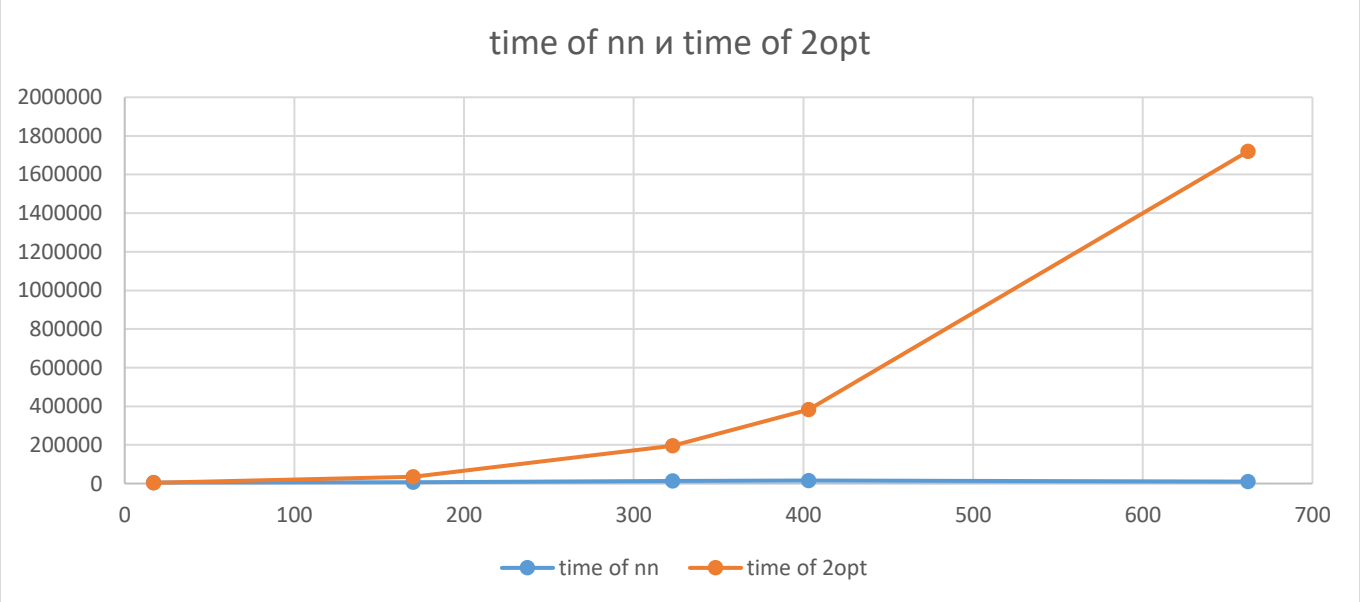
Избираме i = 3 и j = 5

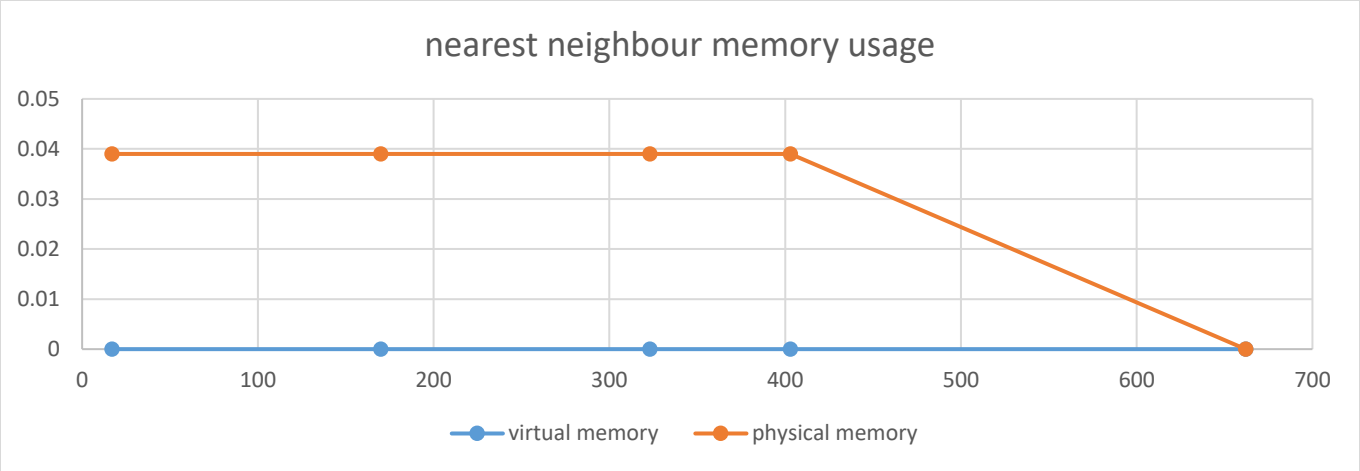
Новият път:

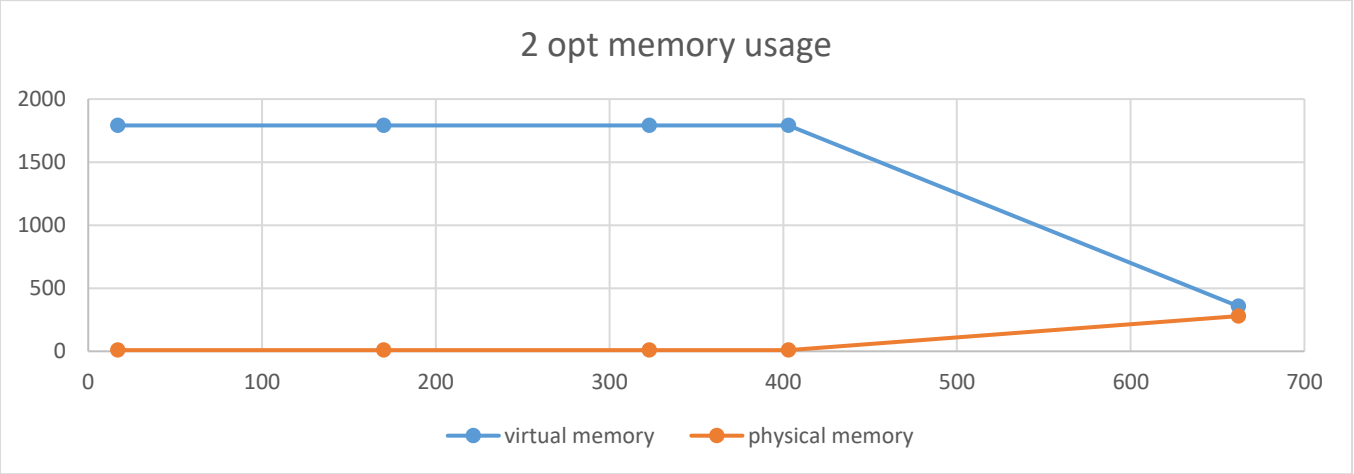
1. A => B
2. A => B => F => E => D => C
3. A => B => F => E => D => C => G => A



Глава 3.Графика на скорост на работа на алгоритмите и заеманата виртуална и физическа памет







Пояснение: Рязката разлика на графика 2 и 3 е от това ,че при последният тест прочитам кординати записвам ги като наредени двойки, съответно смятам разтоянието межу тях и го записвам в матрица. При останалите тестове получавам готова матрица която чета и записвам. Тестовете са 17 града(асиметрични), 170 града(асиметрични), 323 града(асиметрични), 403 града(асиметрични), 662 града(симетрични). Симетричните графи са неориентирани (двупосочни) , а несиметричните са ориентирани (еднопосочни).

Глава 4.Реализация, тесване

4.1. Основната структура от данни която използвам е вектор причината е ,че има доста предимства, като resize, reserve, clear, push_back, shrink_to_fit. За четенето на точките използвам pair с което съхранявам x координати и y

координати. За най-близък съсед използвам стек с цел да държа настоящият град, удобен е да тази цел.

4.2. Тестът с 17 града е избран с цел с по-малкото си градове да се провери дали вярно работят алгоритмите, тестът с 323 града е избран ,за да се покаже как се справя с повече градове, тестът 662 града е избран ,за да се покаже как се справя програмата с подадени координати.

когато се стартира програмата

name of file: br17.atsp

type of file matrix/points: matrix

from what point want to start: 10

изход на програмата: min weigth path is 43 path can be seen in perform_opt.txt

name of file: ftv323.atsp

type of file matrix/points: matrix

from what point want to start: 30

изход на програмата: min weigth path is 3187 path can be seen in perform_opt.txt

name of file: xql662.tsp

type of file matrix/points: points

from what point want to start: 0

изход на програмата: min weigth path is 3060 path can be seen in perform_opt.txt

Използвана литеретура:

Heuristics for the Traveling Salesman Problem by Christian Nilsson.

https://viktorgrigorov.wordpress.com/2017/02/26/290/

https://stackoverflow.com/questions/63166/how-to-determine-cpu-and-memory-consumption-from-inside-a-process

https://www.pluralsight.com/blog/software-development/how-to-measure-execution-time-intervals-in-c--

https://en.wikipedia.org/wiki/2-opt